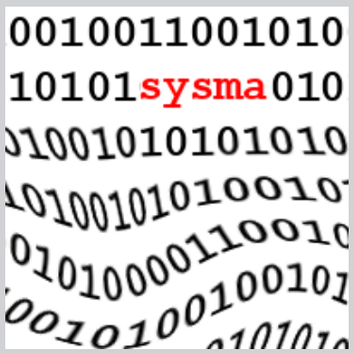
IMT INSTITUTE FOR ADVANCED STUDIES LUCCA

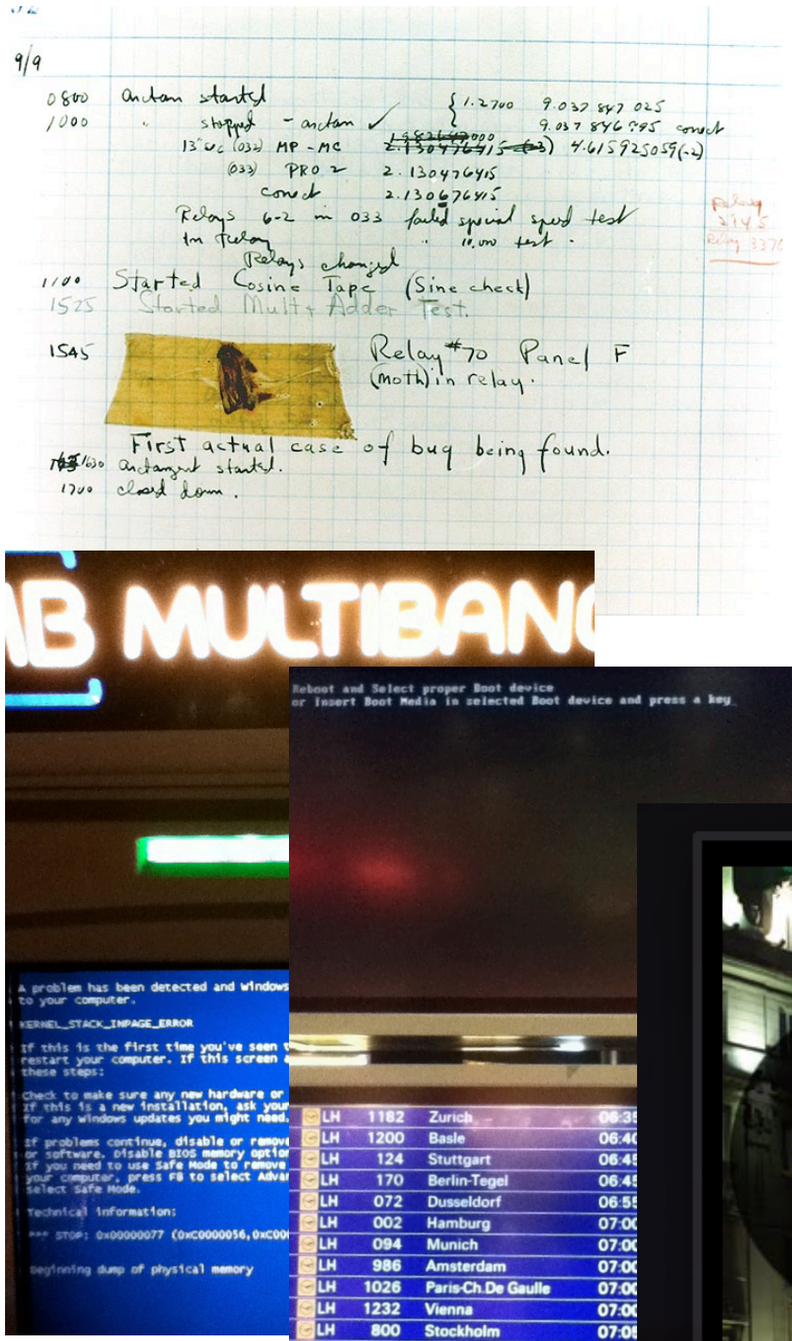
# From Verification to Synthesis of Communication Centered Systems Using Behavioral Types


Hugo Torres Vieira
SysMA - IMT Lucca


London, April 19, 2015

# Software bugs call for verification...





## Is Knight's $440 million glitch the costliest computer bug ever?

By Brian Patrick Eha @CNNTech August 9, 2012: 10:22 AM ET

Recommend 154


## Facebook Fixes Bug That Exposed Private Chats

By Jeff Bertolucci, PCWorld May 5, 2010 12:10 PM

Another day, another Facebook security snafu. The popular social network has patched a major security bug that allowed users to snoop on their friends' private chats, and view their pending friend requests.

The exploit caused Facebook to temporarily disable chat, which was back online as of 11 a.m. (US Pacific) on Wednesday.

Users could inadvertently activate the security breach, which was first reported by TechCrunch Europe, via their privacy settings. Facebook has since patched the bug, and sent *PC World* this statement explaining the mishap:

"For a limited period of time, a bug permitted some users' chat messages and pending friend requests to be made visible to their friends by manipulating the "preview my profile" feature of Facebook privacy settings. When we received reports of the problem, our engineers promptly diagnosed it and temporarily disabled the chat function. We also pushed out a fix to take care of the visible friend requests which is now complete."

Facebook did not report the number of users who were affected by the bug.

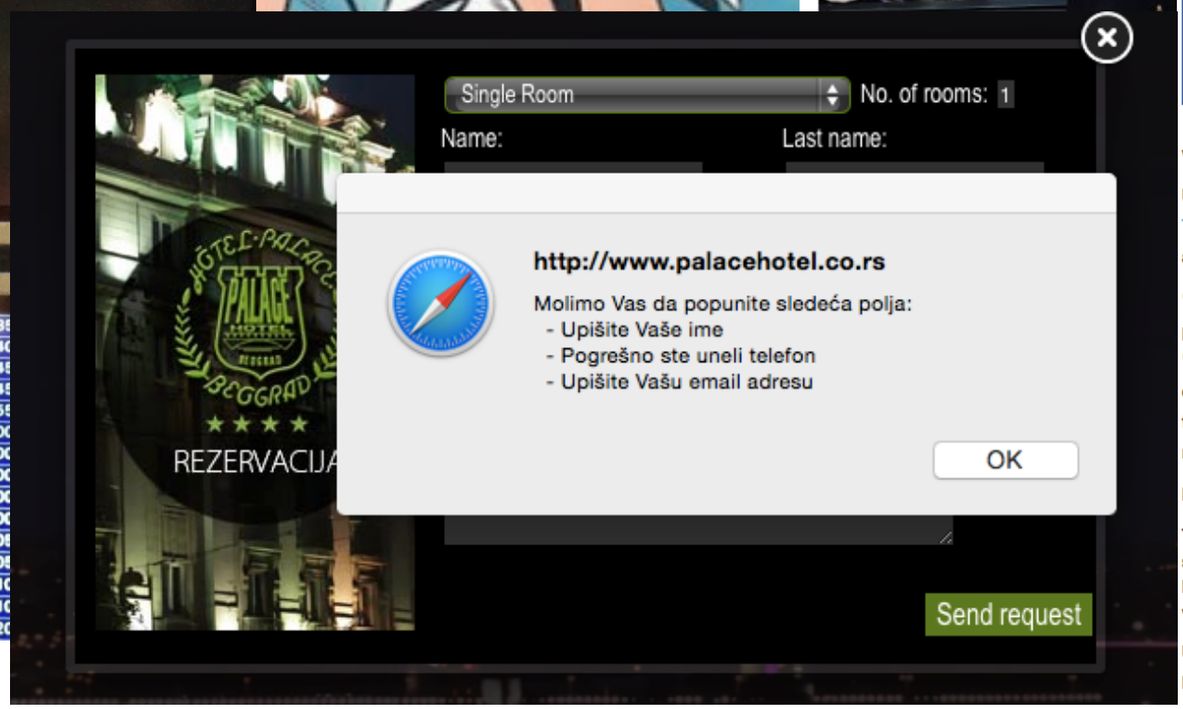
The chat exploit is the latest in a long string of Facebook security controversies. Two months ago, the site was victimized by five different exploits, including four hoax applications and a variant of Koobface virus.

U.S. Senator Charles Schumer (D-N.Y.) last week urged the Federal Trade Commission to create privacy guidelines for









# ... but catching bugs is not enough!


"Don't try to **frighten** people anymore with the terrible things that are going to happen because it is very ineffective"

[Sir Tony Hoare @ MilnerSymposium2012]

(he said he tried for 20 years without success)

- How to disseminate our theories into engineering practice:

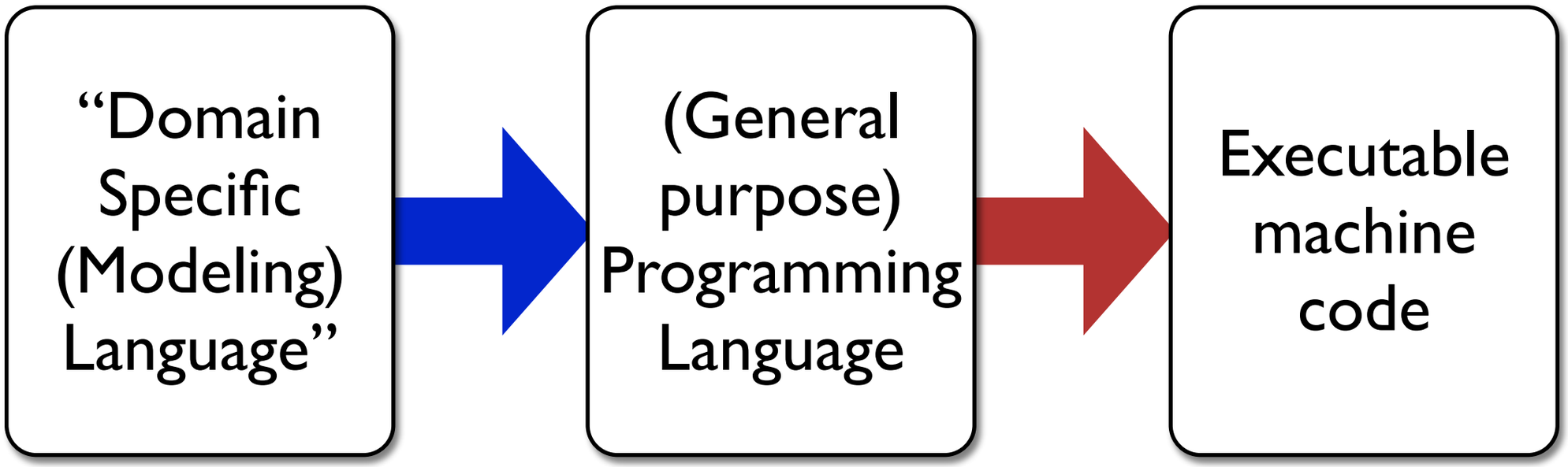
  Develop (e.g., program analysis) tools based on our ideas that allow to make more **effective** and **reliable** software in a **cheaper** way

  Provide ways to **save money, speed up development, and reduce delivery time** is how our methods can be conveyed to practice

# Compilers, PLs, DSLs and beyond

(GP)PLs ⇒ Assembly/Machine code (compilers):

- Ambiguity in (natural language) specs
- Lots of recent work on "correctness"

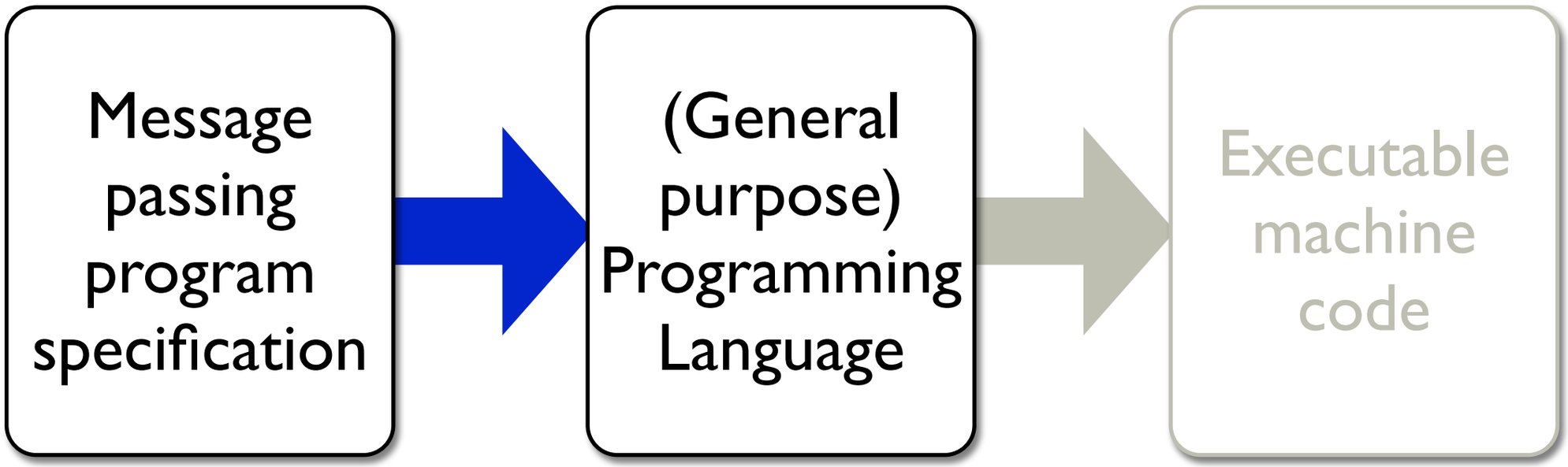



DS(M)Ls -> (GP)PLs

Although DSLs have been widely used to speed up development time there is not so much supporting (theoretical) work

# Communication centered systems

Lots of (theoretical) work on models and specifications




DS(M)Ls -> (GP)PLs

Although DSLs have been widely used to speed up development time there is not so much supporting (theoretical) work

# Derivation, verification and beyond

$$\frac{\vdots \qquad\qquad \vdots}{p_1 \quad \ldots \quad p_k}$$
$$\Gamma \vdash P$$

**Relation** between types (specifications) and programs

# Derivation, verification and beyond

$$\frac{\vdots \qquad\qquad \vdots}{p_1 \quad \ldots \quad p_k}{\Gamma \vdash P}$$

**Relation** between types (specifications) and programs

```
bool type_check(type_spec, program, ...)
```

$$\Gamma \vdash P$$

```
type_spec type_inference(program, ...)
```

$$P \Rightarrow \Gamma \text{ (such that } \Gamma \vdash P\text{)}$$

# Derivation, verification and beyond

$$\frac{\vdots \qquad\qquad \vdots}{p_1 \quad \ldots \quad p_k}{\Gamma \vdash P}$$

**Relation** between types (specifications) and programs

```
bool type_check(type_spec, program, ...)
```

$$\Gamma \vdash P$$

```
type_spec type_inference(program, ...)
```

$$P \Rightarrow \Gamma \text{ (such that } \Gamma \vdash P\text{)}$$

```
program synthesis(type_spec, ...)
```

$$\Gamma \Rightarrow P \text{ (such that } \Gamma \vdash P \text{ - } \textit{Any P?}\text{)}$$

# Progressing Sessions/Conversations

DLY TGC 07

P PLACES 13

Binary

VV COORDINATION 13

BCD et al. CONCUR 08

Multiparty

CV ESOP 09

CDPY COORDINATION 13

PVV COORDINATION 14

# Process Model

$$
\begin{array}{lcll}
P, Q & ::= & \mathbf{0} & \text{(Inaction)} \\
 & | & P \mid Q & \text{(Parallel)} \\
 & | & (\nu x)P & \text{(Restriction)} \\
 & | & x!y.P & \text{(Output)} \\
 & | & x?y.P & \text{(Input)} \\
 & | & \star x?y.P & \text{(Replicated Input)}
\end{array}
$$

# Types

$$
\begin{array}{lcll}
p & ::= & ! & \text{(Output)} \\
 & | & ? & \text{(Input)} \\
 & | & \tau & \text{(Synchronization)} \\
L & ::= & \text{end} & \text{(No interaction)} \\
 & | & \mathsf{e}\ p\ T.\ L & \text{(Session)} \\
T & ::= & L & \text{(Linear)} \\
 & | & \mathsf{e}\ p\ T & \text{(Shared)}
\end{array}
$$

# Typing

- Processes are characterized by their
  - **communication capabilities on each channel**
    - $\Gamma = channel_1 : T_1, ..., channel_k : T_k$
  - **(overall) communication ordering**
    - $<$ = strict partial order of communication events

$$\Gamma ; < \vdash P$$

- the ordering describes the “after than” dependencies among communication events

# Type Theory Results

## Theorem (Preservation)

If $\Gamma;< \vdash P$ and $P \rightarrow Q$ then $\Gamma;< \rightarrow \Gamma';<'$ and $\Gamma';<' \vdash Q$.

## Theorem (Liveness)

Let $\Gamma_1, x: B_1;<_1 \vdash P_1$ and *matched*$(\Gamma_1, x: B_1)$. If e in *events*$(B_1)$ then there is $P_1 \rightarrow^n P_2$ and $(\Gamma_1, x: B_1);<_1 \rightarrow^n (\Gamma_2, x: B_2);<_2$ for some n>0, such that $\Gamma_2, x: B_2;<_2 \vdash P_2$ and e is not in *events*$(B_2)$.

$B_1$, $B_2$ are linear types

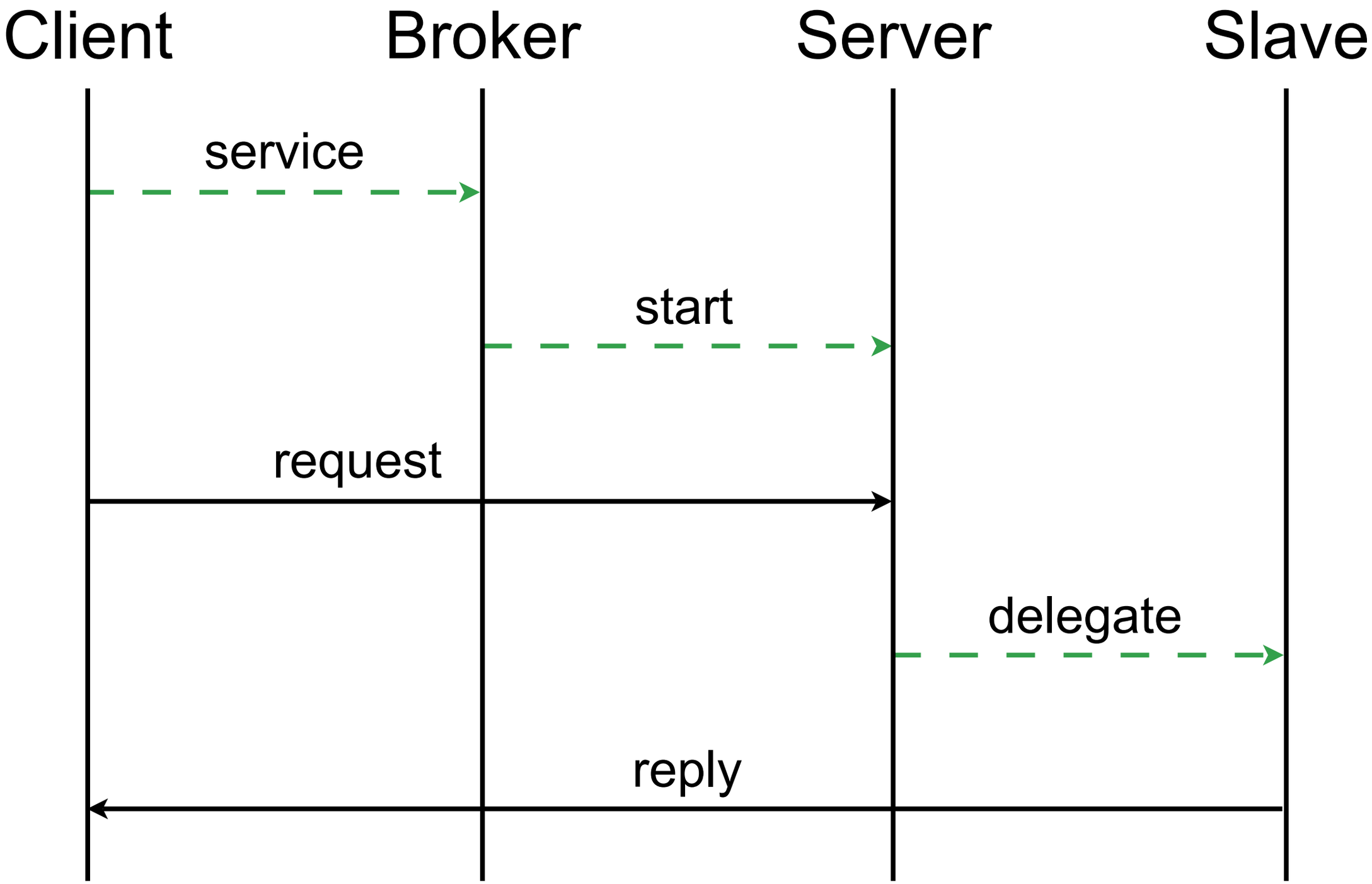
A typing $\Gamma$ is *matched* if (roughly) it only specifies $\tau$ types.

*events*$(B)$ is the set of events mentioned in type $B$

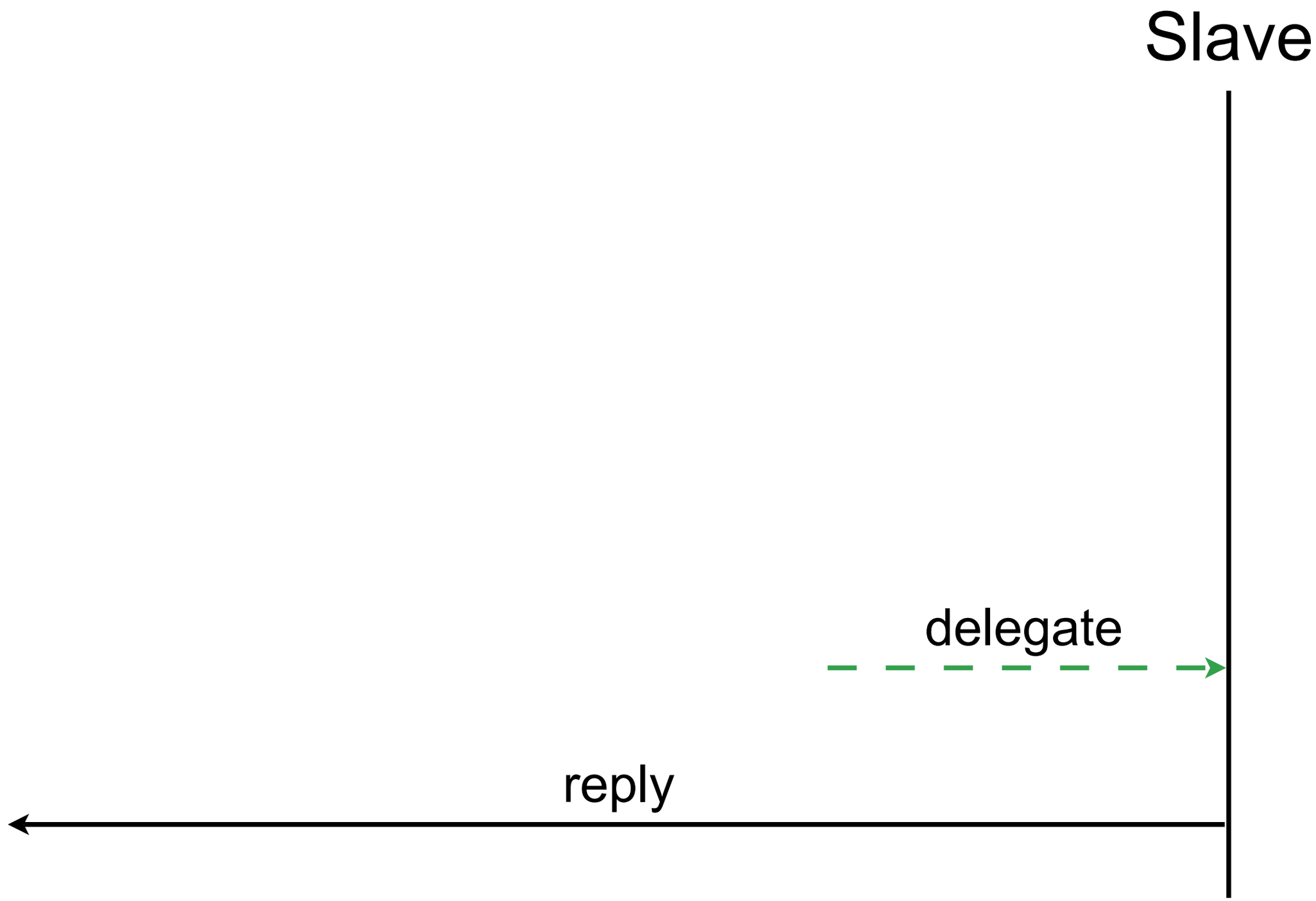
# Synthesis algorithm - Principles

- **Separate synchronization events in the two counterparts**
  - keep relation acyclic considering the unification of the two events but order the individual "polarized" events
- **Explore usages and ordering in a combined way**
  - Synthesize parallel composition whenever order has more than one minimal element
  - If order has one minimal element synthesize the corresponding action prefix
    - continuation is synthesized using events of greater order
    - for outputs if the carried type is "present" in the environment use the respective name, otherwise generate new name (restriction)

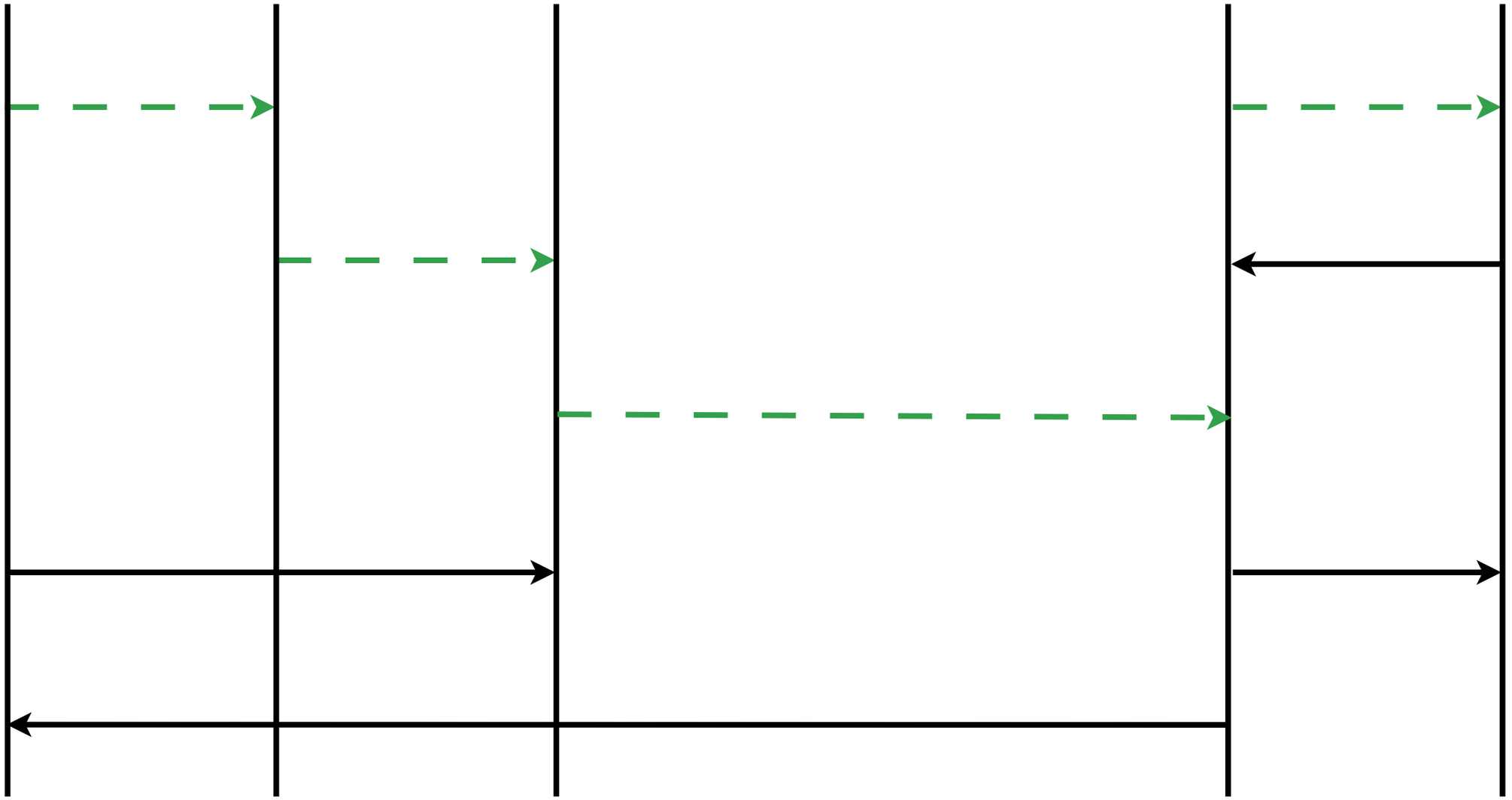
# Message sequence chart - Global view



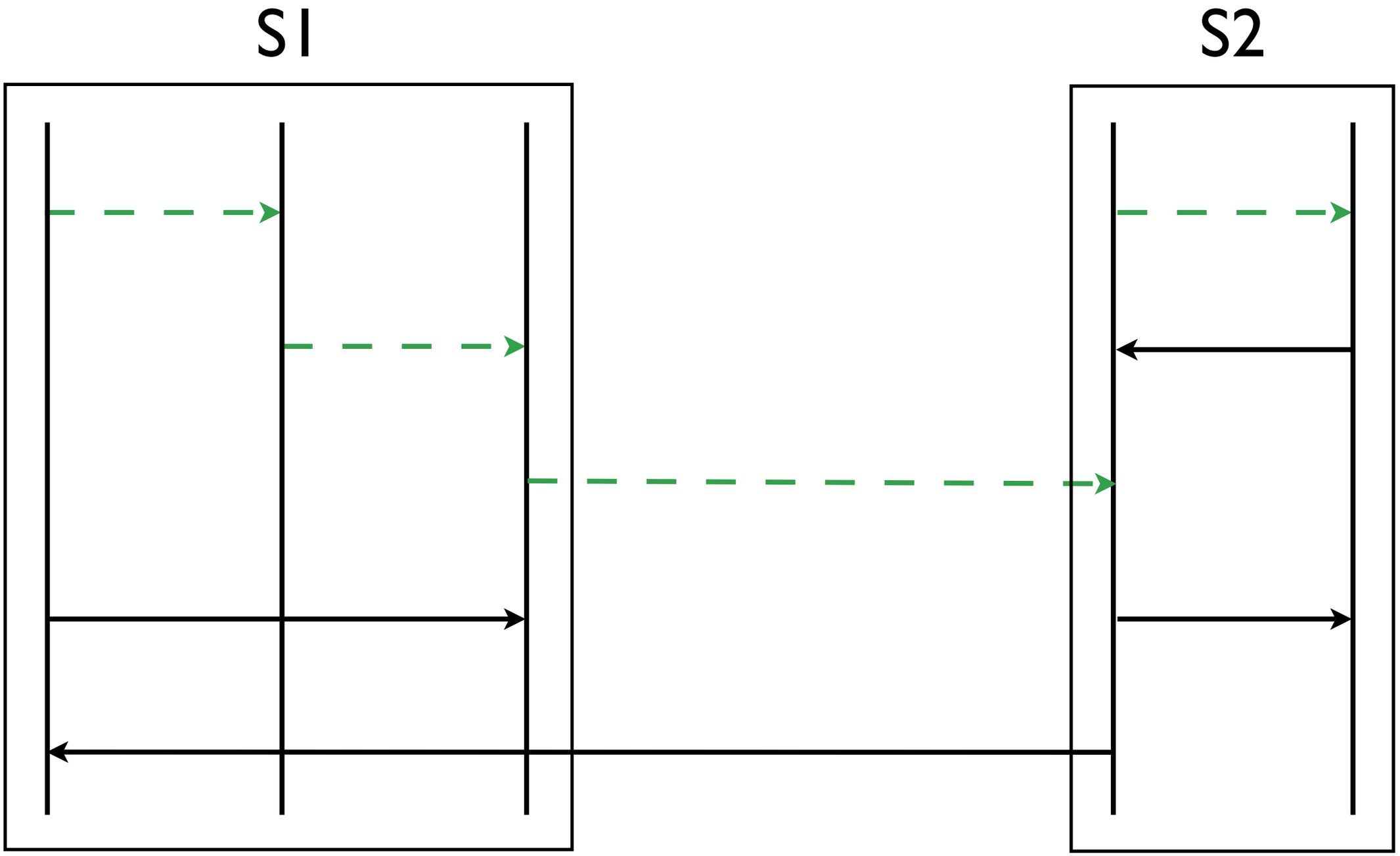
# Message sequence chart - local view



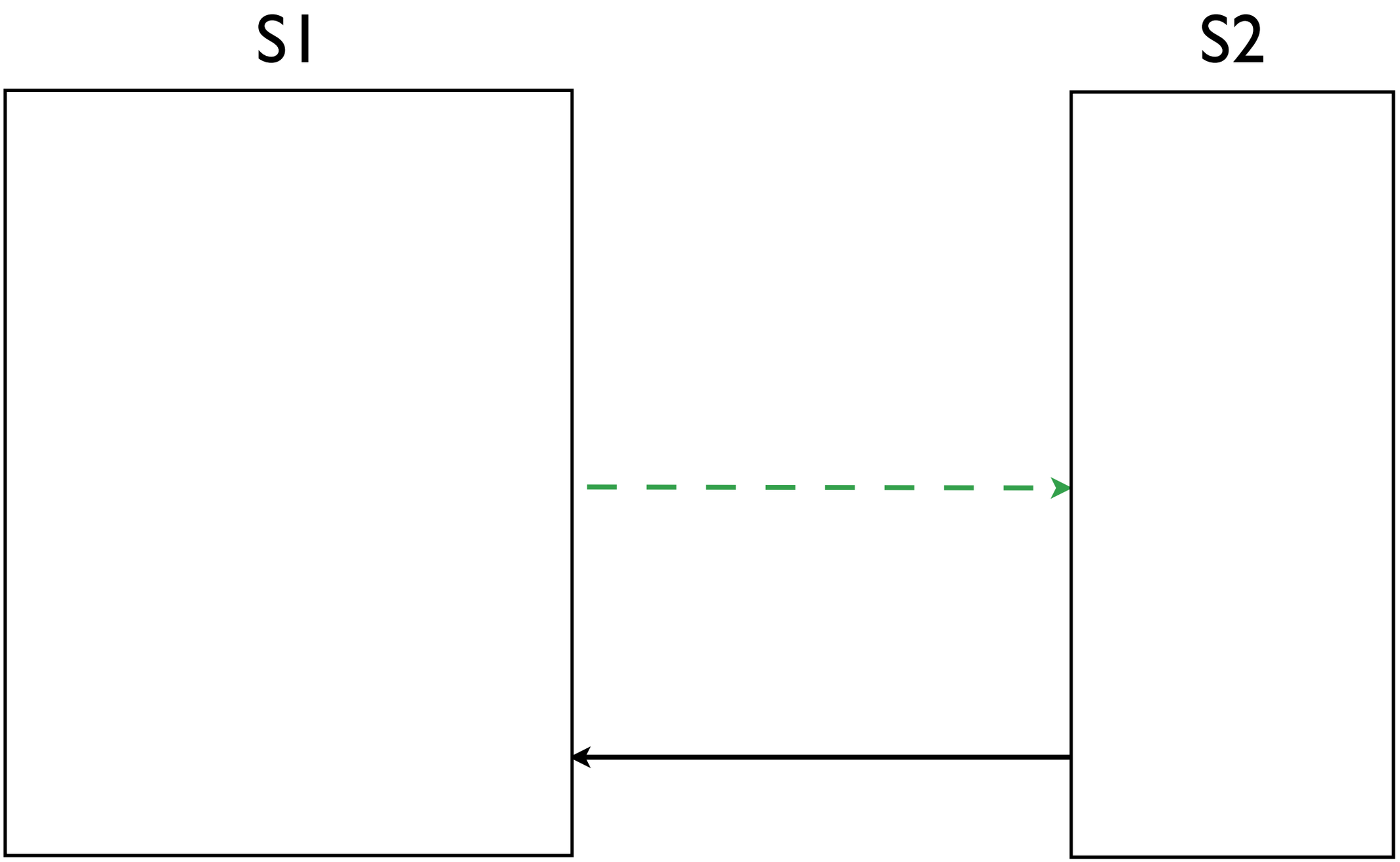
# Session Modules



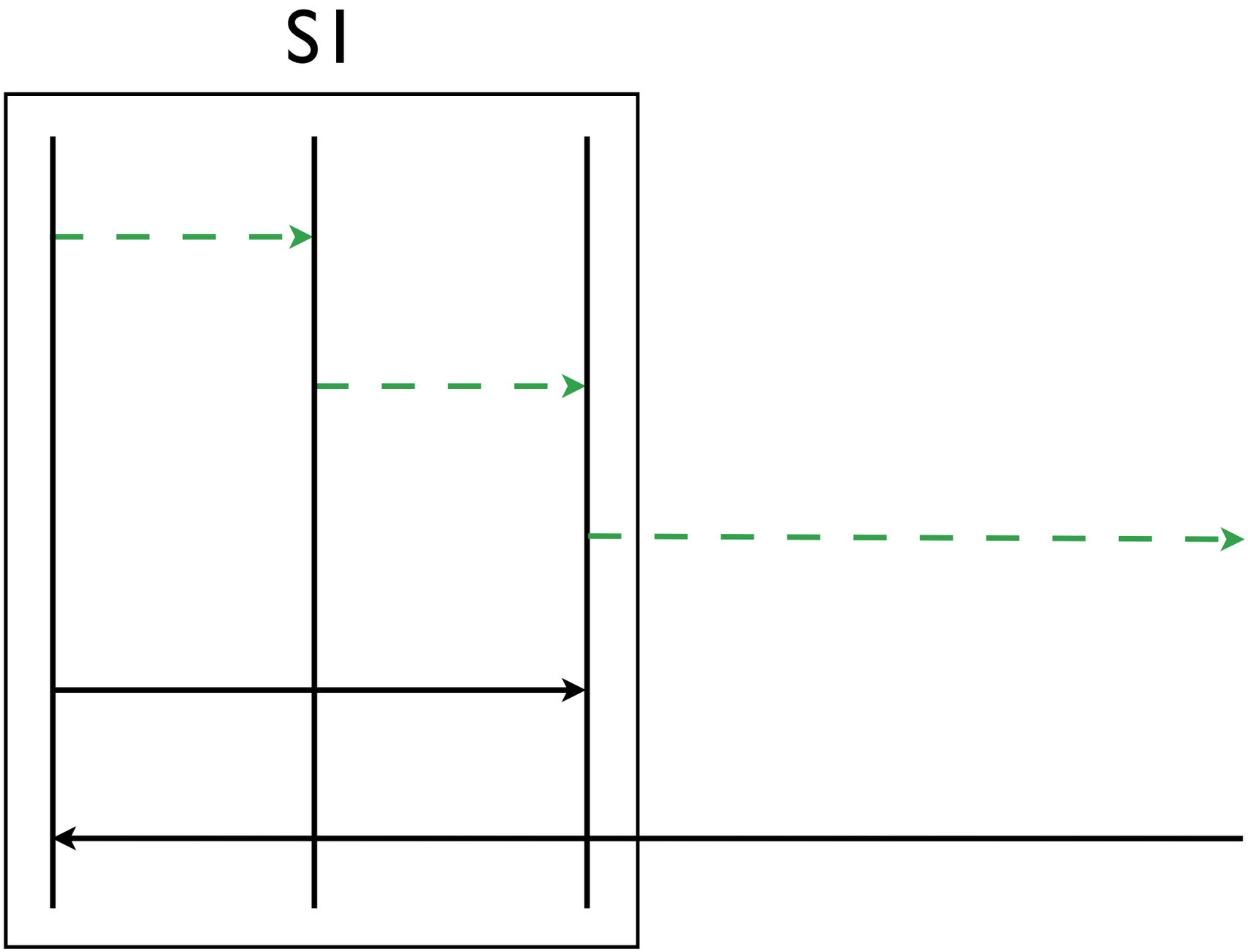
# Session Modules



# Session Modules



# Session Modules



# Advantages of type driven synthesis

- **Type safety and other properties ensured by construction**
  - Preventively saving a lot on debugging time
- **Compositionality**
  - Adding features with compatible specs means new code will be compatible
  - Saving a lot on maintenance effort
- **Validation of generated code wrt specification**
  - For instance liveness provides a tight correspondence between system and spec
  - Behavioral equivalences between specification and system
- **Implementation**
  - Bridge the gap between software engineers and formal specification tools